Voici une série d'exercices basés sur les concepts introduits dans le cours "TP 1 : Les bases" en JavaScript, avec des niveaux progressifs de difficulté pour mieux maîtriser les notions abordées. Ces exercices incluent des corrections pour guider les étudiants.

Exercice 1 : Identifier les types de valeurs

1. Tapez les valeurs suivantes dans la console JavaScript et utilisez `typeof` pour identifier leur type :
    - `42`
    - `true`
    - `'Bonjour'`
    - `[1, 2, 3]`
    - `{ nom: 'Alice', age: 25 }`
    - `undefined`
    - `null`

Correction :

```
typeof 42; // "number"
typeof true; // "boolean"
typeof 'Bonjour'; // "string"
typeof [1, 2, 3]; // "object"
typeof { nom: 'Alice', age: 25 }; // "object"
typeof undefined; // "undefined"
typeof null; // "object" (particularité de JavaScript)
```

Voici quelques exercices pour expliquer les différences entre `var`, `let` et `const`, accompagnés de corrections détaillées :

Exercice 2 : Portée de la variable

Dans cet exercice, nous allons explorer la différence de portée entre `var`, `let` et `const` dans une fonction.

Question :

Écrivez une fonction qui déclare une variable à l'intérieur d'un bloc (`if` ou `for`) avec `var`, `let`, et `const`. Essayez de les accéder en dehors du bloc et observez les résultats.

```
function testPortee() {
    if (true) {
        var varTest = 'Variable avec var';
        let letTest = 'Variable avec let';
        const constTest = 'Variable avec const';
    }

    console.log(varTest); // Qu'affichera cette ligne ?
    console.log(letTest); // Qu'affichera cette ligne ?
    console.log(constTest); // Qu'affichera cette ligne ?
}
```

```
testPortee();
```

Correction :

- **`var`** : La variable `varTest` est accessible en dehors du bloc `if` car `var` a une **portée fonctionnelle** ou **globale**. La sortie sera : `"Variable avec var"`.
- **`let` et `const`** : Les variables `letTest` et `constTest` sont **bloquées** au bloc dans lequel elles ont été déclarées. Elles ne sont pas accessibles en dehors du `if`, ce qui entraînera une erreur de référence. La sortie sera donc :
  - Une erreur pour `letTest` : `ReferenceError: letTest is not defined`
  - Une erreur pour `constTest` : `ReferenceError: constTest is not defined`

Exercice 3 : Redéclaration de variables

Dans cet exercice, nous allons tester la capacité de redéclaration des variables avec `var`, `let`, et `const`.

Question :

Écrivez un programme où vous déclarez une variable avec `var`, `let` et `const`, puis tentez de les redéclarer avec la même syntaxe. Que se passe-t-il dans chaque cas ?

```
var testVar = 'Var initial';
var testVar = 'Var redéclarée'; // Est-ce valide ?

let testLet = 'Let initial';
let testLet = 'Let redéclarée'; // Est-ce valide ?

const testConst = 'Const initial';
const testConst = 'Const redéclarée'; // Est-ce valide ?
```

Correction :

- **`var`** : La redéclaration avec `var` est **possible**. Il n'y a pas d'erreur et la variable est simplement réassignée. La sortie de la variable sera `"Var redéclarée"`.
- **`let`** : Vous ne pouvez pas redéclarer une variable avec `let` dans le même bloc ou la même fonction. Une erreur `SyntaxError` sera générée, comme : `Uncaught SyntaxError: Identifier 'testLet' has already been declared`.
- **`const`** : Il est impossible de redéclarer une variable avec `const`. Comme avec `let`, une erreur `SyntaxError` sera lancée, par exemple : `Uncaught SyntaxError: Identifier 'testConst' has already been declared`.

Exercice 4 : Immutabilité avec `const`

Dans cet exercice, nous allons voir la différence entre `const` et les autres mots-clés concernant l'immutabilité des valeurs.

Question :

Essayez de modifier une variable déclarée avec `const`. Ensuite, modifiez un objet ou un tableau déclaré avec `const`. Quelle différence observiez-vous ?

```
const a = 10;
a = 20; // Tentative de modification de la variable 'a' (quelle erreur
obtenez-vous ?)

const obj = { nom: 'John' };
obj.nom = 'Doe'; // Tentative de modification de la propriété d'un objet
console.log(obj); // Qu'affichez-vous ici ?
```

Correction :

- **`const` avec un type primitif (comme `a`)** : Vous ne pouvez pas réassigner une valeur à une variable déclarée avec `const`. La tentative de réaffecter `a` à `20` entraînera une erreur du type `TypeError: Assignment to constant variable`.
- **`const` avec un objet ou un tableau** : Vous **pouvez** modifier les propriétés d'un objet ou les éléments d'un tableau déclarés avec `const`. Le fait de modifier `obj.nom` de `'John'` à `'Doe'` est parfaitement valide, car seule la **référence** de l'objet est constante, mais ses propriétés peuvent être modifiées. Le `console.log` affichera :

  ```
  { nom: 'Doe' }
  ```

Exercice 5 : Boucles et portée des variables

Explorez comment `var`, `let`, et `const` se comportent dans les boucles, notamment pour l'index de la boucle.

Question :

Écrivez une boucle `for` où l'index est déclaré avec `var`, `let`, et `const`. Essayez d'afficher cet index après la boucle et expliquez le comportement de chaque variable.

```
function testBoucle() {
    for (var i = 0; i < 3; i++) {
        // ...
    }
    console.log(i); // Quelle est la valeur de i ici ?

    for (let j = 0; j < 3; j++) {
        // ...
    }
    console.log(j); // Quelle est la valeur de j ici ?

    for (const k = 0; k < 3; k++) {
        // ...
    }
    console.log(k); // Quelle est la valeur de k ici ?
}

testBoucle();
```

Correction :

- **`var`** : L'index `i` est accessible après la boucle parce que `var` a une portée fonctionnelle ou globale. La valeur de `i` sera `3` (la dernière valeur avant la sortie de la boucle).
- **`let`** : L'index `j` n'est pas accessible en dehors de la boucle, car `let` a une portée de bloc. La tentative d'accès à `j` en dehors de la boucle générera une erreur `ReferenceError`.
- **`const`** : L'index `k` ne peut pas être réassigné dans la boucle, mais le principal problème ici est que la condition `k < 3` ne sera jamais vraie car `k` ne peut pas changer à chaque itération. Cela entraînera une erreur de type `TypeError: Assignment to constant variable`.

Conclusion des exercices :

- **`var`** : Portée fonctionnelle ou globale, possibilité de redéclaration, peut causer des erreurs si mal utilisé.
- **`let`** : Portée de bloc, ne permet pas la redéclaration, plus flexible que `var`.
- **`const`** : Portée de bloc, ne permet pas la redéclaration ni la réassignation de valeurs primitives, mais autorise la modification des propriétés des objets et tableaux.

Exercice 6 : Manipuler les variables

1. Créez une variable `prenom` et affectez-lui votre prénom.
2. Créez une autre variable `age` et affectez-lui votre âge.
3. Affichez un message comme suit dans la console : `"Je m'appelle [prenom] et j'ai [age] ans."` en utilisant la concaténation ou une template string.

Correction :

```
let prenom = 'Alice';
let age = 25;
console.log("Je m'appelle " + prenom + " et j'ai " + age + " ans.");
console.log(`Je m'appelle ${prenom} et j'ai ${age} ans.`);
```

Exercice 7 : Calculs simples

1. Déclarez deux variables, `a` et `b`, avec les valeurs `10` et `20`.
2. Calculez et affichez la somme, la différence, le produit et le quotient de ces deux variables.
3. Modifiez la valeur de `b` pour qu'elle soit égale à `a + 5`.

Correction :

```
let a = 10;
let b = 20;
console.log(a + b); // 30
console.log(a - b); // -10
console.log(a * b); // 200
console.log(a / b); // 0.5
```

```
b = a + 5;
console.log(b); // 15
```

Exercice 8 : Affectations et références

1. Créez un tableau `monTableau = [1, 2, 3]`.
2. Affectez ce tableau à une nouvelle variable `nouveauTableau`.
3. Modifiez le contenu de `monTableau` en remplaçant le deuxième élément par `42`.
4. Que contient `nouveauTableau` ?

Correction :

```
let monTableau = [1, 2, 3];
let nouveauTableau = monTableau;
monTableau[1] = 42;
console.log(monTableau); // [1, 42, 3]
console.log(nouveauTableau); // [1, 42, 3] car les deux variables pointent
vers le même tableau
```

Exercice 9 : Mini-programme interactif

1. Demandez à l'utilisateur de saisir son nom à l'aide de la fonction `prompt()`.
2. Affichez un message personnalisé dans une boîte d'alerte : `"Bonjour [nom] ! Bienvenue dans le monde de JavaScript."`

Correction :

```
let nom = prompt("Quel est votre nom ?");
alert(`Bonjour ${nom} ! Bienvenue dans le monde de JavaScript.`);
```

Exercice 10 : Calculs dynamiques

1. Demandez à l'utilisateur de saisir deux nombres.
2. Convertissez ces entrées en nombres à l'aide de `Number()`.
3. Affichez la somme, le produit et la différence de ces deux nombres dans la console.

Correction :

```
let num1 = Number(prompt("Entrez le premier nombre :"));
let num2 = Number(prompt("Entrez le deuxième nombre :"));

console.log(`Somme : ${num1 + num2}`);
console.log(`Produit : ${num1 * num2}`);
console.log(`Différence : ${num1 - num2}`);
```

Exercice 11 : Résolution d'erreurs

Voici un script avec des erreurs. Corrigez-le :

```
let maVariable = 10
maVariable = "Bonjour
console.log('Le contenu de maVariable est : ' maVariable);
```

Correction :

```
let maVariable = 10;
maVariable = "Bonjour";
console.log('Le contenu de maVariable est : ' + maVariable);
```